SIAM J. SCI. COMPUT.
Vol. 46, No. 4, pp. A2351–A2376

AN ADAPTIVE FACTORIZED NYSTRÖM PRECONDITIONER FOR REGULARIZED KERNEL MATRICES*

SHIFAN ZHAO†, TIANSHI XU†, HUA HUANG‡, EDMOND CHOW‡, AND YUANZHE XI†

Abstract. The spectrum of a kernel matrix significantly depends on the parameter values of the kernel function used to define the kernel matrix. This makes it challenging to design a preconditioner for a regularized kernel matrix that is robust across different parameter values. This paper proposes the adaptive factorized Nyström (`AFN`) preconditioner. The preconditioner is designed for the case where the rank k of the Nyström approximation is large, i.e., for kernel function parameters that lead to kernel matrices with eigenvalues that decay slowly. `AFN` deliberately chooses a well-conditioned submatrix to solve with and corrects a Nyström approximation with a factorized sparse approximate matrix inverse. This makes `AFN` efficient for kernel matrices with large numerical ranks. `AFN` also adaptively chooses the size of this submatrix to balance accuracy and cost.

Key words. kernel matrices, preconditioning, sparse approximate inverse, Nyström approximation, farthest point sampling, Gaussian process regression

MSC codes. 65F08, 65F10, 65F55, 68W25

DOI. 10.1137/23M1565139

See reproducibility of computational results at end of the article.

1. Introduction.

In this paper, we seek efficient preconditioning techniques for the iterative solution of large, regularized linear systems associated with a kernel matrix $\mathbf{K}$,

$$(\mathbf{K} + \mu\mathbf{I})\,\mathbf{a} = \mathbf{b}, \tag{1.1}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix, $\mu \in \mathbb{R}$ is a regularization parameter, $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, and $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the kernel matrix whose (i,j)th entry is defined as $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ with a symmetric positive semidefinite (SPSD) kernel function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and data points $\{\mathbf{x}_i\}_{i=1}^n$. A function $\mathcal{K}(\cdot,\cdot)$ defined over a domain $\mathcal{X}$ is SPSD if $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathcal{K}(\mathbf{x}_j, \mathbf{x}_i)$, $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ and $\sum_{i=1}^n \sum_{j=1}^n c_i c_j \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \geq 0$, $\forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ given $n \in \mathbb{N}$ and $c_1, c_2, \ldots, c_n \in \mathbb{R}$. For example, $\mathcal{K}$ can be chosen as a Gaussian kernel function,

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{l^2}\right), \tag{1.2}$$

where l is a kernel function parameter called the *length-scale.*

Linear systems of the form (1.1) appear in many applications, including kernel ridge regression (KRR) [1] and Gaussian process regression (GPR) [37]. When the

*Submitted to the journal's Numerical Algorithms for Scientific Computing section April 12, 2023; accepted for publication (in revised form) April 9, 2024; published electronically July 17, 2024.
https://doi.org/10.1137/23M1565139
Funding: The research of the first, second, and fifth authors is supported by NSF award OAC 2003720. The second author is also partially supported by NSF award DMS 1912048. The research of the third and fourth authors is supported by NSF award OAC 2003683.
†Department of Mathematics, Emory University, Atlanta, GA 30322 USA (shifan.zhao@emory.edu, tianshi.xu@emory.edu, yxi26@emory.edu).
‡School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332-4017 USA (huangh223@gatech.edu, echow@cc.gatech.edu).

number of data points n is small, solution methods based on dense matrix factorizations are the most efficient. When n is large, a common approach is to solve (1.1) using a sparse or low-rank approximation to $\mathbf{K}$ [39, 40, 34]. In this paper, we pursue an exact solution approach for (1.1) with iterative methods. Fast matrix-vector multiplications by $\mathbf{K}$ for the iterative solver are available through fast transforms [23, 54] and hierarchical matrix methods [3, 7, 18, 9, 2, 13, 38, 43, 31]. This paper specifically addresses the problem of preconditioning for the iterative solver.

In KRR, GPR, and other applications, the kernel function parameters must be estimated to fit the data at hand. This involves an optimization process, for example, maximizing a likelihood function, which in turn involves solving (1.1) for kernel matrices given the same data points but different values of the kernel function parameters. Different values of the kernel function parameters lead to different characteristics of the kernel matrix. For the Gaussian kernel function above, Figure 1 (left) shows the eigenvalue spectrum of 61 regularized 1000×1000 kernel matrices. 1000 data points are sampled inside a cube with edge length 10. In all the experiments, the side of the d-dimensional cube is scaled by $n^{1/d}$ in order to maintain a constant density as we increase the number of data points. Figure 1 (right) demonstrates the significant variation in the number of iterations required by the unpreconditioned conjugate gradient (CG) method to solve linear systems associated with these kernel matrices. Notably, linear systems are more efficiently solved at the extremes of l, whether very high or very low, compared to moderate values. This is consistent with the theoretical convergence results of CG. The convergence of CG is fast, for example, when there exists a cluster of eigenvalues with only a few outliers deviating from this cluster. In the context of kernel matrices, a small l yields a kernel matrix that closely resembles a diagonal matrix which has a tight cluster of eigenvalues. On the other hand, a large l typically results in a scenario where, aside from a few outliers, most eigenvalues are around the regularization parameter μ.

In this paper, we seek a preconditioner for kernel matrix systems (1.1) that is adaptive to different kernel matrices $\mathbf{K}$ corresponding to different values of kernel function parameters such that it can flatten the curve in Figure 1 (right). When the numerical rank of $\mathbf{K}$ is small, there exist good methods [42, 20] for preconditioning

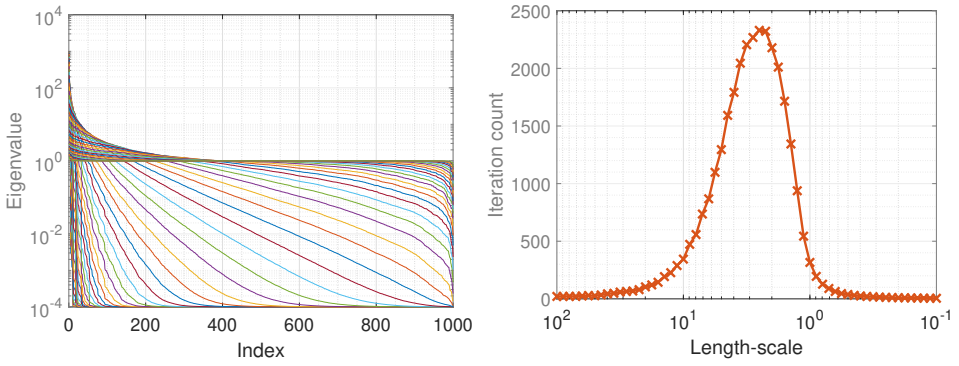

FIG. 1. *Left: Spectrum of* 61 *regularized Gaussian kernel matrices associated with the same* 1000 *points sampled randomly over a cube with edge length* 10 *and a fixed regularization parameter* $\mu = 0.0001$ *but different length-scales* l. *Right: Iteration counts of unpreconditioned CG to solve equation* (1.1) *for the* 61 *regularized kernel matrices to reach the relative residual tolerance* 10^{-4}.

$\mathbf{K}+\mu\mathbf{I}$ based on a Nyström approximation [52] to the kernel matrix. We will provide a self-contained description of the Nyström approximation in section 2, as it is related to our proposed preconditioner.

2. Background: Nyström approximation. The Nyström approximation, originally developed for approximating the eigenfunctions of integral operators, has emerged as an indispensable technique for generating low-rank approximations of kernel matrices. For a kernel matrix $\mathbf{K}\in\mathbb{R}^{n\times n}$, the Nyström approximation takes the following form [32]:

$$\mathbf{K}_{\text{nys}}=\mathbf{K}\mathbf{X}(\mathbf{X}^\top\mathbf{K}\mathbf{X})^\dagger(\mathbf{K}\mathbf{X})^\top, \tag{2.1}$$

where $\mathbf{X}\in\mathbb{R}^{n\times k}$ is a test matrix, and † denotes the pseudoinverse. The choice of $\mathbf{X}$ significantly influences the Nyström approximation accuracy and efficiency. Employing $\mathbf{X}$ as a submatrix of a permutation matrix facilitates the sampling of k columns from $\mathbf{K}$, and $\mathbf{X}^\top\mathbf{K}\mathbf{X}$ yields a $k\times k$ submatrix. This sampling-based Nyström approximation offers a geometric insight when examining a kernel matrix over a dataset $X=\{\mathbf{x}_i\}_{i=1}^n$, equating the sampling of k columns from $\mathbf{K}$ to the selection of k points from X. These points, referred to as landmark points, are crucial for the accuracy of the approximation. Let X_k represent the k samples from the dataset X, and assume the $k\times k$ submatrix is invertible, (2.1) can be rewritten as

$$\mathbf{K}_{\text{nys}}=\mathbf{K}_{X,X_k}\mathbf{K}_{X_k,X_k}^{-1}\mathbf{K}_{X_k,X}, \tag{2.2}$$

where $\mathbf{K}_{X,Y}$ encapsulates $[\mathcal{K}(x,y)]_{x\in X,\,y\in Y}$ for any two datasets X and Y. With the landmark points indexed first, $\mathbf{K}$ can be partitioned into a block 2-by-2 matrix,

$$\mathbf{K}=\begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12}\\ \mathbf{K}_{12}^\top & \mathbf{K}_{22}\end{bmatrix}, \tag{2.3}$$

where $\mathbf{K}_{11}=\mathbf{K}_{X_k,X_k}$, $\mathbf{K}_{12}=\mathbf{K}_{X_k,X\setminus X_k}$, and $\mathbf{K}_{22}=\mathbf{K}_{X\setminus X_k,X\setminus X_k}$. Consequently, the sampling-based Nyström approximation approximates $\mathbf{K}_{22}$ with $\mathbf{K}_{12}^\top\mathbf{K}_{11}^{-1}\mathbf{K}_{12}$:

$$\mathbf{K}_{\text{nys}}=\begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12}\\ \mathbf{K}_{12}^\top & \mathbf{K}_{12}^\top\mathbf{K}_{11}^{-1}\mathbf{K}_{12}\end{bmatrix}. \tag{2.4}$$

This strategy underscores the importance of landmark point selection in formulating an effective approximation. A different approach to constructing Nyström approximations utilizes the projection method where alternative test matrices, such as Gaussian random matrices, are utilized in this context. However, the projection-based Nyström approximation incurs a higher computational cost due to explicit matrix multiplications, unlike its sampling counterpart. Nevertheless, this method has stronger theoretical foundations, and the approximation error can be close to that of the best rank-k approximation attainable via singular value decomposition (SVD) [32]. Note that direct implementation of projection-based Nyström approximation following equation (2.1) may result in numerical instability. As a result, the following form is recommended [20]:

$$\mathbf{K}_{\text{nys}}=\mathbf{U}\widehat{\mathbf{\Lambda}}\mathbf{U}^\top, \tag{2.5}$$

where $\mathbf{U}$ comprises orthonormal columns, and $\widehat{\mathbf{\Lambda}}$ is a diagonal matrix. The details for constructing $\mathbf{U}$ and $\widehat{\mathbf{\Lambda}}$ can be found in [20]. When employing Nyström approximation as a preconditioner for preconditioned CG (PCG) to solve (1.1), in conjunction with

the Sherman–Morrison–Woodbury (SMW) formula and the orthonormality of $\mathbf{U}$, the inverse of the preconditioner $\mathbf{U}\widehat{\mathbf{\Lambda}}\mathbf{U}^\top + \mu\mathbf{I}$ can be expressed as

$$(\mathbf{U}\widehat{\mathbf{\Lambda}}\mathbf{U}^\top + \mu\mathbf{I})^{-1} = \mathbf{U}(\widehat{\mathbf{\Lambda}} + \mu\mathbf{I})^{-1}\mathbf{U}^\top + \frac{1}{\mu}(\mathbf{I} - \mathbf{U}\mathbf{U}^\top). \tag{2.6}$$

Equation (2.6) is valid for any low-rank approximation of $\mathbf{K}$, provided $\mathbf{U}$ is orthonormal. While applying the SMW formula directly to equation (2.2) based on the sampling-based Nyström preconditioner seems plausible, this implementation is not numerically stable. Therefore, (2.6) is preferred for both sampling-based and projection-based Nyström preconditioners. Preconditioners leveraging Nyström approximations and other low-rank approaches for the kernel matrix $\mathbf{K}$ typically involve eigendecomposition or another factorization of a dense $k \times k$ matrix, which is feasible for small k but becomes prohibitively expensive for large k, where k is often set to be the numerical rank. The numerical rank is defined as $\#\{\lambda_i > c\mu \mid \lambda_i \text{ are the eigenvalues of } \mathbf{K} + \mu\mathbf{I}, \forall i \in [n]\}$, where $\#\{\cdot\}$ denotes the count of eigenvalues exceeding $c\mu$. This definition bears resemblance to the concept of the effective dimension defined as $\text{Trace}(\mathbf{K}(\mathbf{K} + c\mu)^{-1}) = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda_i + c\mu}$ [20]. However, we found that the numerical rank definition better serves our purpose of adaptively estimating the rank. In section 3, we propose a block 2-by-2 approximate factorization of $\mathbf{K} + \mu\mathbf{I}$ as a preconditioner, with the (1,1) block corresponding to the set of landmark points. Additionally, a method for estimating the number of landmark points is presented in section 3.3. The selection of landmark points based on farthest point sampling (FPS) is supported by the analysis in section 4. The efficiency of the proposed preconditioner is demonstrated in section 5, while section 6 summarizes this paper's contributions.

3. Adaptive factorized Nyström preconditioner. This section outlines the development of the proposed preconditioner, initially motivated by the screening effect [44, 45, 46] and the `FSAI` (factorized sparse approximate inverse) method [28]. Building on these concepts, we introduce the adaptive factorized Nyström (`AFN`) preconditioner, designed to enhance computational efficiency through a factorized matrix representation. To enhance performance across different kernel parameters, we also propose a rank estimation technique.

3.1. Screening effect and FSAI. In Gaussian processes, the kernel matrix $\mathbf{K}_{XX}$ is used to characterize the covariance of the target values $\mathbf{y} = \{y_i\}_{i=1}^n$ of the data points $X = \{\mathbf{x}_i\}_{i=1}^n$. Since $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$ can be considered as the conditional covariance matrix of target values at $X \backslash X_k$ conditioned on those at X_k, the screening effect [24, 36, 44, 45, 46, 40, 39] in geostatistics implies that optimal linear predictions at a point in a Gaussian process primarily rely on nearby data points. While the theory provides specific conditions for this effect, it is practically leveraged to improve the computational efficiency of Gaussian process regression. The Vecchia approximation [47], rooted in this concept, simplifies joint density calculations by conditioning on neighboring points, leading to a sparse Cholesky factorization of the precision matrix. However, the approximation accuracy depends on the strength of the screening effect and the number of neighboring points considered. More specifically, the screening effect suggests that the optimal linear prediction of the target value y_i at a point $\mathbf{x}_i$ in a Gaussian process typically depends on the values at neighboring points $N_i = \{\mathbf{x}_j | j \in D_i\}$, where D_i are the indices of the nearest neighbors of $\mathbf{x}_i$ which can be determined using the K-nearest neighbors (KNN) algorithm. This has been theoretically scrutinized, and conditions for its validity have been established, albeit under limited scenarios [44, 45, 46]. The Vecchia approximation [47] utilizes this principle

Algorithm 3.1 Factorized sparse approximate inverse (`FSAI`).

1. **Input**: A symmetric positive definite matrix $\mathbf{K}$, the number of nonzeros on each row w, a lower triangular sparsity pattern $\mathbf{S} \in \mathbb{R}^{n\times n}$ which is a zero-one matrix with at most w ones on each row indicating the positions of w-nearest neighbors of each data point.
2. **for** $i = 1$ to n **do**
3. Extract the nonzero pattern $\mathbf{s}_i$ from the ith row of $\mathbf{S}$ with length $w \ll n$
4. Compute $\mathbf{G}_{i,\mathbf{s}_i} = \frac{\mathbf{e}_w^\top (\mathbf{K}_{\mathbf{s}_i,\mathbf{s}_i})^{-1}}{\sqrt{\mathbf{e}_w^\top (\mathbf{K}_{\mathbf{s}_i,\mathbf{s}_i})^{-1}\mathbf{e}_w}}$
5. **end for**
6. **Return**: $\mathbf{G}$

by approximating the exact joint density $p_1(\mathbf{y}) = p(y_1)\prod_{i=2}^n p(y_i|y_{1:i-1}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ with $p_2(\mathbf{y}) = p(y_1)\prod_{i=2}^n p(y_i|y_{N_i}) \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{K}})$, significantly simplifying calculations. This approximation yields a precision matrix $\hat{\mathbf{K}}^{-1}$ with a sparse Cholesky decomposition, where the Cholesky factor has a limited number of nonzero entries per row, equal to the size of N_i [14, 26]. While recent studies [40, 39] confirm that the screening effect is valid for functions derived from Green's functions of elliptic operators, it is important to note that when the effect is weak or absent, the approximation will not be very accurate. Despite its theoretical limitations, the Vecchia approximation has recently achieved empirical success in numerous applications [39, 26, 24]. Remarkably, this approximation is equivalent to the factorized sparse approximate inverse (`FSAI`) method introduced by Kolotilina and Yeremin [28]. Therefore, based on the screening effect, we can use `FSAI` to compute a sparse approximate inverse $\mathbf{G}$ of the lower triangular Cholesky factor of a symmetric positive definite (SPD) matrix $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{21}(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$, given a sparsity pattern $\mathbf{S}$ for $\mathbf{G}$, i.e., $\mathbf{G}^\top\mathbf{G} \approx (\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{21}(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$. An important feature of `FSAI` is that the computation of $\mathbf{G}$ only requires the entries of $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{21}(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$ corresponding to the sparsity pattern of $\mathbf{G}$ and $\mathbf{G}^\top$. This makes it possible to economically compute $\mathbf{G}$ even if $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{21}(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$ is large and dense. Further, the computation of each row of $\mathbf{G}$ is independent of other rows, and thus the rows of $\mathbf{G}$ can be computed in parallel. The nonzero pattern used for row i of $\mathbf{G}$ corresponds to the $w-1$ nearest neighbors of point i that are numbered less than i (since $\mathbf{G}$ is lower triangular), where w is a parameter which controls the nonzeros per row. The pseudocode of `FSAI` can be found in Algorithm 3.1.

In the next subsection, we will discuss how to apply `FSAI` on the $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$ to improve the robustness and accuracy of the Nyström approximation.

3.2. AFN preconditioner construction and application. We now propose a new preconditioner for $\mathbf{K} + \mu\mathbf{I}$ that can be efficiently constructed even when k is large. Recall that $\mathbf{K}_{11}$ is the kernel matrix associated with a set of landmark points X_k. To control the computational cost, we impose a limit on the maximum size of X_k, setting it to a constant value, such as 2000. For any SPD matrix $\mathbf{A}$, the Cholesky decomposition ensures that the matrix can be factored as $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$, where $\mathbf{L}$ is a lower triangular matrix. Let $\mathbf{L}\mathbf{L}^\top$ be the Cholesky factorization of $\mathbf{K}_{11}+\mu\mathbf{I}$ and $\mathbf{G}^\top\mathbf{G}$ be the `FSAI` of $(\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^\top(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$. Then we can define the following factorized preconditioner for $\mathbf{K} + \mu\mathbf{I}$:

$$\mathbf{M} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{K}_{12}^\top\mathbf{L}^{-\top} & \mathbf{G}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{L}^\top & \mathbf{L}^{-1}\mathbf{K}_{12} \\ \mathbf{0} & \mathbf{G}^{-\top} \end{bmatrix}. \tag{3.1}$$

Algorithm 3.2 Adaptive factorized Nyström (`AFN`) preconditioner construction.

1. **Input**: Dataset $X = \{\mathbf{x}_i\}_{i=1}^n$ for $\mathbf{x}_i \in \mathbb{R}^d$, Kernel function $\mathcal{K}(\cdot,\cdot)$, regularization parameter μ, estimated rank k returned by Algorithm 3.4.
2. **if** $d \le 10$ **then**
3. Select k landmark points denoted by X_k according to FPS Algorithm 4.1.
4. **else**
5. Select k landmark points denoted by X_k using uniform sampling.
6. **end if**
7. Perform Cholesky factorization: $\mathbf{L} = \mathrm{Chol}(\mathbf{K}_{11} + \mu\mathbf{I})$ where $\mathbf{K}_{11} = \mathcal{K}(X_k, X_k)$
8. Invoke Algorithm 3.1 to compute $\mathbf{G} = \mathrm{FSAI}(\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12})$ where $\mathbf{K}_{12} = \mathcal{K}(X_k, X_r)$, $\mathbf{K}_{22} = \mathcal{K}(X_r, X_r)$, $X_r = X \setminus X_k$.
9. **Return**: Matrices $\mathbf{L}$ and $\mathbf{G}$

Expanding the factors,

$$\mathbf{M} = \begin{bmatrix} \mathbf{K}_{11} + \mu\mathbf{I} & \mathbf{K}_{12} \\ \mathbf{K}_{12}^\top & (\mathbf{G}^\top\mathbf{G})^{-1} + \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12} \end{bmatrix} \tag{3.2}$$

$$= \mathbf{K}_{nys} + \mu\mathbf{I} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\mathbf{G}^\top\mathbf{G})^{-1} + \mathbf{K}_{12}^\top\left((\mathbf{K}_{11} + \mu\mathbf{I})^{-1} - (\mathbf{K}_{11})^{-1}\right)\mathbf{K}_{12} - \mu\mathbf{I} \end{bmatrix}}_{\text{Correction term}}, \tag{3.3}$$

we see that $\mathbf{M}$ equals $\mathbf{K}_{nys} + \mu\mathbf{I}$ plus a correction term. Thus the preconditioner is not a Nyström preconditioner but has similarities to it. Unlike a Nyström preconditioner, the factorized form approximates $\mathbf{K} + \mu\mathbf{I}$ entirely and does not approximate $\mathbf{K}$ separately, and thus it avoids the SMW formula. In particular, when we have $\mathbf{G}^\top\mathbf{G} = (\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$ exactly, $\mathbf{M} = \mathbf{K} + \mu\mathbf{I}$.

We call the preconditioner defined in (3.1) the adaptive factorized Nyström (`AFN`) preconditioner and summarize its construction procedure in Algorithm 3.2. Owing to the factorization structure, it is sufficient to store $\mathbf{L}$ and $\mathbf{G}$ from Algorithm 3.2.

The preconditioning operation for `AFN` solves systems with the matrix $\mathbf{M}$. Assuming that the vectors $\mathbf{r}$ and $\mathbf{s}$ are partitioned conformally with the block structure of $\mathbf{M}$, in order to solve the system

$$\mathbf{M}\begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix},$$

the algorithm is

$$\begin{aligned} \mathbf{s}_2 &:= \mathbf{G}^\top\mathbf{G}\left(\mathbf{r}_2 - \mathbf{K}_{12}^\top(\mathbf{L}^{-\top}\mathbf{L}^{-1})\mathbf{r}_1\right), \\ \mathbf{s}_1 &:= \mathbf{L}^{-\top}\mathbf{L}^{-1}(\mathbf{r}_1 - \mathbf{K}_{12}\mathbf{s}_2). \end{aligned}$$

The pseudocode detailing the application of the `AFN` preconditioner is described in Algorithm 3.3.

In the next two subsections, we will discuss two pivotal components in the construction of `AFN`: the selection of an optimal number of landmark points and their identification method. Addressing these components is essential for refining the Nyström approximation and, by extension, the efficacy of the `AFN` method. The choice of landmark points, specifically the determination of the block size of $\mathbf{K}_{11}$, plays a critical

Algorithm 3.3 Adaptive factorized Nyström (`AFN`) preconditioner application.

1. **Input**: Vector $\mathbf{r}$, matrices $\mathbf{L}$, $\mathbf{G}$, $\mathbf{K}_{12}$
2. Partition $\mathbf{r}$ conformally with the size of $\mathbf{L}$ and $\mathbf{G}$ as $[\mathbf{r}_1, \mathbf{r}_2]^\top$
3. Solve $(\mathbf{K}_{11} + \mu\mathbf{I})\mathbf{z} = \mathbf{r}_1$ by computing $\mathbf{z} = \mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{r}_1$
4. Compute $\mathbf{s}_2 = \mathbf{G}^\top\mathbf{G}(\mathbf{r}_2 - \mathbf{K}_{12}^\top\mathbf{z})$
5. Solve $(\mathbf{K}_{11} + \mu\mathbf{I})\mathbf{s}_1 = (\mathbf{r}_1 - \mathbf{K}_{12}\mathbf{s}_2)$ by computing $\mathbf{s}_1 = \mathbf{L}^{-\top}\mathbf{L}^{-1}(\mathbf{r}_1 - \mathbf{K}_{12}\mathbf{s}_2)$
6. **Return**: Vector $\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2]^\top$

role. An underestimation may lead to inadequate approximations, whereas an overestimation could introduce unnecessary computational burdens and potential numerical instability. To navigate this challenge, we propose Algorithm 3.4 for empirically estimating the optimal number of landmark points, a process elaborated in section 3.3. It is also equally crucial to devise a strategy for selecting these landmark points. For low-dimensional data, we advocate for FPS due to its balance of simplicity, efficiency, and effectiveness, a rationale further explored in section 4. Conversely, for datasets with high-dimensional features, we recommend uniform sampling as a means to alleviate the computational cost.

3.3. Adaptive choice of approximation rank. To construct a preconditioner that is adaptive and efficient for a range of regularized kernel matrices arising from different values of the kernel function parameters, it is necessary to estimate the rank of the kernel matrix $\mathbf{K}$. For example, if the estimated rank is small enough that it is inexpensive to perform an eigendecomposition of a k-by-k matrix, then the Nyström preconditioner should be used due to the reduced construction cost.

It is of course too costly in general to use a rank-revealing decomposition of $\mathbf{K}$ to compute k. Instead, we will compute k that approximately achieves a certain Nyström approximation accuracy via checking the relative Nyström approximation error on a subsampled dataset.

First, a dataset X_m of m points is randomly subsampled from X. The number of points m is an input to the procedure, and m can be much smaller than the k that will be computed. Then the coordinates of the data points in X_m are scaled by $(m/n)^{1/d}$, and the smaller kernel matrix $\mathbf{K}_{X_m,X_m}$ is formed. The rationale of this scaling is that we expect the spectrum of $\mathbf{K}_{X_m,X_m}$ to have a similar decay pattern as that of $\mathbf{K}_{X,X}$. We now run FPS on X_m to construct Nyström approximations with increasing rank to $\mathbf{K}$ until the relative Nyström approximation error falls below 0.1, and we define this Nyström rank as r. Finally, we approximate the Nyström rank of $\mathbf{K}$ as rn/m. Figure 2 plots the Nyström approximation errors on subsampled matrices and original matrices associated with two different length-scales. The data points X are generated randomly by sampling 1000 points uniformly within a cube, and $m = 100$ points are subsampled randomly. The two relative Nyström approximation error curves show a close match in both cases. This rank estimation method is summarized in Algorithm 3.4. We also find that if the estimated rank is small (e.g., less than 2000), we can perform an eigen-decomposition of $\mathbf{K}_{X_m,X_m}$ associated with the unscaled data points and refine the estimation with the number of eigenvalues greater than 0.1μ. Here we estimate the numerical rank defined in the previous section with $c = 0.1$ to have a refined rank estimation to further enhance performance.

If the estimated rank k is smaller than 2000, then the Nyström preconditioner should be used. `AFN` is only constructed when the estimated rank exceeds 2000 for better efficiency. The selection of the preconditioning method is shown precisely in Algorithm 3.5.

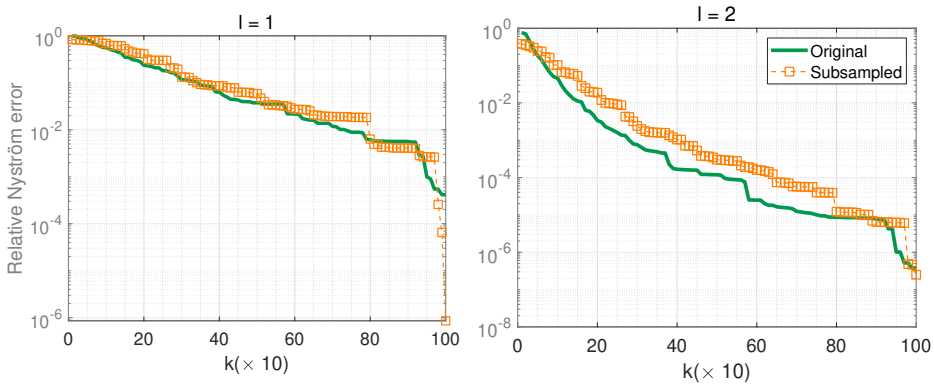

FIG. 2. *Comparison of the relative Nyström approximation error curves for an original dataset and a subsampled dataset with* 100 *points, associated with two different length-scales. The original dataset contains* 1000 *uniformly sampled points from a cube with edge length* 10. *The indices of the subsampled dataset are matched with those of the original dataset by computing the relative Nyström approximation errors on the original dataset only for ranks that are multiples of* 10. *The plot shows how the approximation error changes as the rank of the approximation increases.*

Algorithm 3.4 Nyström rank estimation.

1. **Input**: Dataset X with size n, subsample size m, and kernel function $\mathcal{K}(\mathbf{x}, \mathbf{y})$
2. **Output**: Approximate Nyström rank k
3. Randomly subsample a subset X_m of m points from X and scale the coordinates of X_m by $(m/n)^{1/d}$
4. Form the $m \times m$ matrix $\mathbf{K}_{X_m, X_m}$
5. Find the Nyström rank r such that the relative Nyström approximation error for $\mathbf{K}_{X_m, X_m}$ with FPS sampling implemented in Algorithm 4.1 falls below 0.1
6. **if** $k \geq 2000$ **then**
7. **Return** $k = rn/m$
8. **else**
9. Compute eigenvalues of $\mathbf{K}_{X_m, X_m}$ associated with the unscaled data points
10. Return k, the number of eigenvalues greater than 0.1μ
11. **end if**

The choice of the landmark points affects the accuracy of the overall `AFN` preconditioner, just as this choice affects the accuracy of the Nyström preconditioners. The sparsity and the conditioning of $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top (\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$ generally improves when more landmark points are chosen, which would, on the other hand, increase the computational cost and the instability of the Cholesky factorization of $\mathbf{K}_{11} + \mu\mathbf{I}$. Finally, we combine every component and summarize the PCG with `AFN` in Algorithm 3.5. In the next section, the choice of landmark points is discussed in light of these considerations.

4. Selecting the landmark points. Existing methodologies for sampling k landmark points from a dataset with n data points include uniform sampling [52], the anchor net method [8, 10], leverage score sampling [16, 21, 34], k-means-based sampling [55], determinantal point process (DPP)-based sampling [4], and random pivoted Cholesky sampling [12]. Uniform sampling with the computational complexity $O(k)$ excels in scenarios such as kernel ridge regression applications, where direct access to kernel matrices is available, and the data does not exhibit unbalanced clus-

Algorithm 3.5 Preconditioned conjugate gradient with the proposed preconditioning scheme.

1. **Input**: Kernel matrix $\mathbf{K}$, regularization parameter μ, right-hand side vector $\mathbf{b}$
2. Estimate numerical rank k of $\mathbf{K}$ with Algorithm 3.4
3. **if** $k \geq 2000$ **then**
4. Solve $(\mathbf{K} + \mu\mathbf{I})\mathbf{a} = \mathbf{b}$ using PCG with the `AFN` preconditioner, applied as per Algorithm 3.3
5. **else**
6. Solve $(\mathbf{K} + \mu\mathbf{I})\mathbf{a} = \mathbf{b}$ using PCG with the column sampling-based Nyström preconditioner, applied as per equation (2.6)
7. **end if**
8. **Return**: approximate solution vector

ters. Nonetheless, its efficacy diminishes when faced with unbalanced clusters, as it tends to oversample larger clusters. To address this shortcoming, adaptive sampling techniques have been proposed. These methods, including leverage score sampling, DPP-based sampling, and random pivoted Cholesky sampling, employ nonuniform sampling distributions derived from kernel matrices. For instance, ridge leverage score sampling constructs the probability for sampling the ith column proportional to the ith diagonal entry of $(\mathbf{K} + \mu\mathbf{I})^{-1}\mathbf{K}$. In [34], a recursive sampling strategy was introduced, reducing the computational cost of ridge leverage score sampling to $O(nk)$ kernel evaluations and $O(nk^2)$ running time. k-DPP-based sampling extends the sampling distribution across all k-subsets of $1, \ldots, n$, albeit at a much higher computational cost of $O(n^3)$. However, a Markov chain Monte Carlo (MCMC) approach proposed in [30] can reduce this cost to linear time under some conditions. Due to the challenges of verifying these conditions and the necessity of reevaluating a $k \times k$ determinant, k-DPP-based sampling has experienced limited acceptance in practice compared to other sampling methods. Random pivoted Cholesky sampling, as presented in [12], introduces a method aligned with pivoted Cholesky procedures, where the ith pivot is selected proportional to the magnitude of the diagonal entries of the Schur complement at the ith step. This method necessitates $O(n(k+1))$ kernel evaluations. Geometry-based sampling is another avenue, with k-means sampling clustering data points into k clusters and utilizing the centroids as landmark points at a cost of $O(tkn)$, where t represents the iteration count in Lloyd's algorithm. The anchor net method [8], an efficient tactic to mitigate the limitations of uniform sampling in high-dimensional datasets, employs a low-discrepancy sequence to diminish gaps and clusters compared to uniform sampling, while maintaining robust space coverage, at a complexity of $O(nk)$. In our proposed preconditioner, a few different sampling methods can be employed. We opt for farthest point sampling (FPS) due to its simplicity, ease of use, cost-effectiveness, and independence from the length-scale parameter. Specifically, landmark points will be selected based on a balance between two geometric measures to ensure the preconditioner's effectiveness and robustness.

The first measure h_{X_k}, called *fill distance* [19, 29], is used to quantify how well the points in X_k fill out a domain Ω as follows:

$$h_{X_k} = \max_{\mathbf{x} \in \Omega \setminus X_k} \operatorname{dist}(\mathbf{x}, X_k), \tag{4.1}$$

where $\operatorname{dist}(\mathbf{x}, Y) = \inf_{\mathbf{y} \in Y} \|\mathbf{x} - \mathbf{y}\|$ is the distance between a point $\mathbf{x}$ and a set Y, and where Ω denotes the domain of the kernel function under consideration, which can

be either a continuous region or a finite discrete set. The geometric interpretation of this measure is the largest radius of an empty ball in Ω that does not intersect with X_k. This implies that X_k with a smaller fill distance will better fill out Ω.

The second measure q_{X_k}, called *separation distance* [19, 29], is defined as the distance between the closest pair of points in X_k:

$$q_{X_k} = \min_{\mathbf{x}_{k_i}, \mathbf{x}_{k_j} \in X_k, k_i \neq k_j} \operatorname{dist}(\mathbf{x}_{k_i}, \mathbf{x}_{k_j}). \tag{4.2}$$

The geometric interpretation of this measure is the diameter of the largest ball that can be placed around every point in X_k such that no two balls overlap. A larger q_{X_k} indicates that the columns in $\mathbf{K}_{11}$ tend to be more linearly independent and thus leads to a more well-conditioned $\mathbf{K}_{11}$. Given that the separation distance serves as a metric for the conditioning of the kernel matrix [49], and the conditioning of $\mathbf{K}_{11}$ will affect the numerical stability of $\mathbf{L}$, a larger separation distance implies a more stable Nyström approximation and a more stable `AFN` preconditioner.

As more landmark points are sampled, both h_{X_k} and q_{X_k} tend to decrease. We wish to choose X_k such that h_{X_k} is small, while q_{X_k} is large. We will analyze the interplay between h_{X_k} and q_{X_k} in section 4.1. In particular, we will show that if $h_{X_k} \leq C q_{X_k}$ for some constant C, then h_{X_k} and q_{X_k} have the same order as the minimal value of the fill distance and the maximal value of the separation distance that can be achieved with k points, respectively.

Moreover, we find that FPS [17] can generate landmark points with $h_{X_k} \leq q_{X_k}$. FPS is often used in mesh generation [35] and computer graphics [41]. In spatial statistics, FPS is also known as MaxMin ordering (MMD) [24]. FPS initializes X_1 with an arbitrary point $\mathbf{x}_{k_1}$ in X (better choices are possible). At step $i+1$, FPS selects the point that is farthest away from X_i,

$$\mathbf{x}_{k_{i+1}} = \arg\max_{\mathbf{x} \in X \setminus X_i} \operatorname{dist}(\mathbf{x}, X_i). \tag{4.3}$$

See Figure 3 for an illustration of FPS on a two-dimensional dataset, and see the complete pseudocode of FPS in Algorithm 4.1. The landmark points, selected through FPS, are distributed evenly across the dataset, avoiding the formation of dense clusters. This property will be analyzed in the following subsections.

4.1. Interplay between fill and separation distance. In this section, we will study the relationship between h_{X_k} and q_{X_k}. We will show that if $h_{X_k} \leq C q_{X_k}$ for a constant C, then h_{X_k} and q_{X_k} will have the same order as the minimal fill distance and maximal separation distance that can be achieved with any subset with k points, respectively.

First, notice that there exist a lower bound for h_{X_k} and an upper bound for q_{X_k}, which is analyzed in the next theorem when all the points are inside a unit ball in $\mathbb{R}^d$.

THEOREM 4.1. *Suppose all the data points are inside a unit ball Ω in $\mathbb{R}^d$. Then for an arbitrary subset $X_k = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\}$ of X, the following bounds hold for h_{X_k} and q_{X_k}:*

$$h_{X_k} \geq k^{-1/d} \quad \textit{and} \quad q_{X_k} \leq 6k^{-1/d}. \tag{4.4}$$

Proof. In order to show the lower bound of h_{X_k}, we first derive an upper bound of the volume of Ω. Notice that $\Omega \subset \bigcup_{i=1}^{k} B_{h_{X_k}}(\mathbf{x}_{k_i})$, where $B_{h_{X_k}}(\mathbf{x}_{k_i})$ is the ball centered at $\mathbf{x}_{k_i}$ with radius h_{X_k}. Then

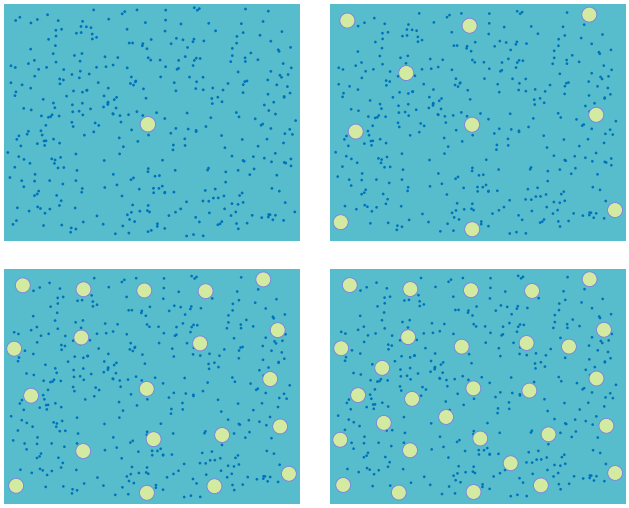

FIG. 3. *An illustration of FPS for selecting* 1, 10, 20, *and* 30 *points from a two-dimensional dataset with* 400 *points where the big circles represent the selected points and the dots denote the other data points.*

$$\mathrm{Vol}(\Omega) \le \sum_{i=1}^{k} \mathrm{Vol}(B_{h_{X_k}}(\mathbf{x}_{k_i})) = k \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} h_{X_k}^d.$$

This gives us the first bound.

Similarly, we get an upper bound of q_{X_k} by using the packing number bound from [53]:

$$k \le \Big(\frac{3}{\frac{q_{X_k}}{2}}\Big)^d.$$

Thus we have

$$q_{X_k} \le 6k^{-\frac{1}{d}}.$$

This gives us the second bound. ∎

Remark 4.2. When Ω satisfies the interior cone condition [33], similar bounds $h_{X_k} \ge C_\Omega k^{-1/d}$ and $q_{X_k} \le C'_\Omega k^{-1/d}$ can be derived for more complex bounded domains, where C_Ω and C'_Ω are two constants depending on the domain Ω.

The above bounds show that the minimal fill distance h_{X_k} cannot be smaller than $k^{-1/d}$, while the maximal separation distance q_{X_k} cannot be greater than $6k^{-1/d}$ and $\frac{1}{6} q_{X_k} \le h_{X_k}$ when the domain is a unit ball in $\mathbb{R}^d$. In the following theorem, we show that if a sampling scheme can select a subset X_k with $h_{X_k} \le C q_{X_k}$, then q_{X_k} has the same order as the maximal separation distance that can be achieved by a subset with k points.

THEOREM 4.3. *Assume the data points are on a bounded domain* Ω *that satisfies the interior cone condition; then if* $h_{X_k} \le C q_{X_k}$,

$$C_\Omega k^{-1/d} \le h_{X_k} \le C \times C'_\Omega k^{-1/d}, \quad \frac{C_\Omega}{C} k^{-1/d} \le q_{X_k} \le C'_\Omega k^{-1/d}. \tag{4.5}$$

Proof. If $h_X \le Cq_X$, then we have

$$C_\Omega k^{-1/d} \le h_{X_k} \le Cq_{X_k} \le C \times C'_\Omega k^{-1/d}. \qquad \square$$

Theorem 4.3 shows that h_{X_k} is at most $C \times \frac{C'_\Omega}{C_\Omega}$ times larger than its theoretical lower bound, and q_{X_k} is at least $\frac{1}{C} \times \frac{C_\Omega}{C_{\Omega'}}$ times as large as its theoretical upper bound in this case.

4.2. Optimal properties of FPS. Although FPS is a greedy algorithm designed to select a set of data points with maximal dispersion at each iteration, it can generate X_k with h_{X_k} at most two times the minimal fill distance [22] and q_{X_k} at least half the largest separation distance over all subsets with k points [51]. In the next theorem, we first confirm that FPS can generate X_k satisfying $h_{X_k} \le q_{X_k}$ and then demonstrate its two near-optimality properties in a cohesive manner. While these properties have been independently established in [22, 51], our analysis amalgamates and revalidates these results within a unified framework.

THEOREM 4.4. *Suppose the minimal fill distance of a subset with k points is achieved with X_k^*, and the maximal separation distance of a subset with k points is achieved with X_{k*}. Then the set X_k sampled by FPS satisfies*

$$h_{X_k} \le q_{X_k} \quad \textit{and} \quad q_{X_k} \ge \frac{1}{2} q_{X_{k*}} \quad \textit{and} \quad h_{X_k} \le 2h_{X_k^*}. \tag{4.6}$$

Proof. Without loss of generality, we assume the subset X_k sampled by FPS contains the points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. Suppose $q_{X_k} = \operatorname{dist}(\mathbf{x}_j, \mathbf{x}_m)$ with $j < m < (k+1)$, and point $\mathbf{x}_m$ is selected at iteration m by FPS; then

$$h_{X_{m-1}} = \max_{\mathbf{x} \in X \setminus X_{m-1}} \operatorname{dist}(\mathbf{x}, X_{m-1}) = \operatorname{dist}(\mathbf{x}_j, \mathbf{x}_m) = q_{X_k}. \tag{4.7}$$

Since h_{X_k} is a nonincreasing function of k, we have $h_{X_k} \le h_{X_{m-1}} = q_{X_k}$.

We now prove $q_{X_k} \ge \frac{1}{2} q_{X_k *}$. According to the definition, there exists a subset with k points $X_{k*} = \{\mathbf{x}_*^1, \ldots, \mathbf{x}_*^k\}$ such that

$$q_{X_{k*}} = \max_{Y \subset X, |Y| = k} \min_{\mathbf{x}_i, \mathbf{x}_j \in Y} \operatorname{dist}(\mathbf{x}_i, \mathbf{x}_j).$$

According to (4.7), we know that all the points in X must lie in one of the $m-1$ disks defined by

$$C(\mathbf{x}_i, q_{X_k}) = \{\mathbf{x} | \|\mathbf{x} - \mathbf{x}_i\| \le q_{X_k}\}, \quad i \in [m-1]. \tag{4.8}$$

Since $m - 1 < k$, at least two points $\mathbf{x}_*^i, \mathbf{x}_*^j \in X_{k*}$ must belong to the same disk centered at some $\mathbf{x}_l$. Therefore, $2q_{X_k} \ge \operatorname{dist}(\mathbf{x}_*^i, \mathbf{x}_l) + \operatorname{dist}(\mathbf{x}_*^j, \mathbf{x}_l) \ge \operatorname{dist}(\mathbf{x}_*^i, \mathbf{x}_*^j) \ge q_{X_k *}$ via the triangle inequality.

Next, we prove $h_{X_k} \le 2h_{X_k^*}$. At the kth iteration of FPS, the set X can be split into k clusters $\{C_i\}_{i=1}^k$ such that the point $\mathbf{x}$ in X will be classified into cluster C_i if $\operatorname{dist}(\mathbf{x}_i, \mathbf{x}) \le \operatorname{dist}(\mathbf{x}_j, \mathbf{x})$, $\forall j \ne i$. At the $(k+1)$th iteration of FPS, one more point $\mathbf{x}_{k+1}$ will be selected. Then we can show that

$$\operatorname{dist}(\mathbf{x}_i, \mathbf{x}_j) \ge h_{X_k} \quad \text{for} \quad i, j \in \{1, 2, \ldots, k+1\},$$

and in particular,

$$q_{X_{k+1}} \ge h_{X_k}.$$

Assume $\mathbf{x}_{k+1} \in C_i$. From the definition of h_{X_k}, we know that $\text{dist}(\mathbf{x}_{k+1}, \mathbf{x}_i) = h_{X_k}$ and $\text{dist}(\mathbf{x}_{k+1}, \mathbf{x}_j) \geq \text{dist}(\mathbf{x}_{k+1}, \mathbf{x}_i)$ for $j \neq i$. Moreover, we have $\text{dist}(\mathbf{x}_i, \mathbf{x}_j) \geq q_{X_k}$ for $j \neq k+1$. Since $q_{X_k} \geq h_{X_k}$, we know $q_{X_{k+1}} = \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \geq h_{X_k}$.

Finally, assume $\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_k^*$ is the optimal subset of X that achieves the minimal fill distance with cardinality k. Now the set X can be split into k clusters $\{C_i^*\}_{i=1}^k$ such that the point $\mathbf{x}$ in X will be classified into C_i^* if $\text{dist}(\mathbf{x}_i^*, \mathbf{x}) \leq \text{dist}(\mathbf{x}_j^*, \mathbf{x})$, $\forall j \neq i$. Assume the points selected by FPS in the first $k+1$ iterations are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k+1}$. We know that at least two points from $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k+1}$ belong to the same cluster. Denote these two points as $\mathbf{x}_p$ and $\mathbf{x}_q$, and the corresponding cluster is C_j^*. Then we have

$$h_{X_k} \leq q_{X_{k+1}} \leq \text{dist}(\mathbf{x}_p, \mathbf{x}_q) \leq \text{dist}(\mathbf{x}_p, \mathbf{x}_i^*) + \text{dist}(\mathbf{x}_q, \mathbf{x}_i^*) \leq 2h_{X_k^*},$$

which indicates that $h_{X_k} \leq 2h_{X_k^*}$. ∎

4.3. Nyström approximation error analysis based on fill distance. Now we justify why FPS is a good landmark points selection method from the perspective of Nyström approximation error. Define the Nyström approximation error as

$$\|\mathbf{K} - \mathbf{K}_{nys}\| = \|\mathbf{K}_{22} - \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{K}_{12}\|.$$

In this section, we will show that the Nyström approximation error is also related to the fill distance h_{X_k}. In particular, for Gaussian kernels defined in (1.2) and inverse multiquadric kernels

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = (c^2 + \|\mathbf{x} - \mathbf{y}\|^2)^{-\frac{p}{2}},\ p > 0,\ c \in \mathbb{R}, \tag{4.9}$$

we can derive a Nyström approximation error estimate in terms of the fill distance, as presented in the following theorem.

THEOREM 4.5. *The Nyström approximation* $\mathbf{K}_{nys} = \mathbf{K}_{X,X_k}\mathbf{K}_{X_k,X_k}^{-1}\mathbf{K}_{X_k,X}$ *to* $\mathbf{K}$ *using the landmark points* $X_k = \{\mathbf{x}_{k_i}\}_{i=1}^k$ *has the error estimate*

$$\|\mathbf{K} - \mathbf{K}_{nys}\| < \sqrt{n\|\mathbf{K}\|}C' \exp(-C''/h_{X_k}), \tag{4.10}$$

where C' *and* C'' *are constants independent of* X_k *but dependent on kernels and the domain.*

The detailed proof of Theorem 4.5 is in Appendix A. This theorem is a discrete version of Theorem A in [5], which implies that kernel operators corresponding to smooth kernels are effectively low rank. Our proof adapts the original results on kernel functions, as presented in [5], to discrete matrix settings. This extension shows that the low-rank approximation mentioned in [5] can indeed be interpreted as a Nyström approximation applicable to matrices. For this Nyström approximation. Theorem 4.5 implies that landmark points X_k with a smaller fill distance can yield a more accurate Nyström approximation. We illustrate this numerically with an experiment. In Figure 4, we plot the fill distance curve and the Nyström approximation error curve corresponding to a Gaussian kernel with $l = 10$ when 1000 points are uniformly sampled from a cube with edge length 10. We test random sampling and FPS for selecting the landmark points and observe that FPS leads to a smaller fill distance than random sampling. We also observe that FPS Nyström can achieve lower approximation errors than the randomly sampled one when the same k is used. Thus we will use FPS to select landmark points in the construction of Nyström-type preconditioners if

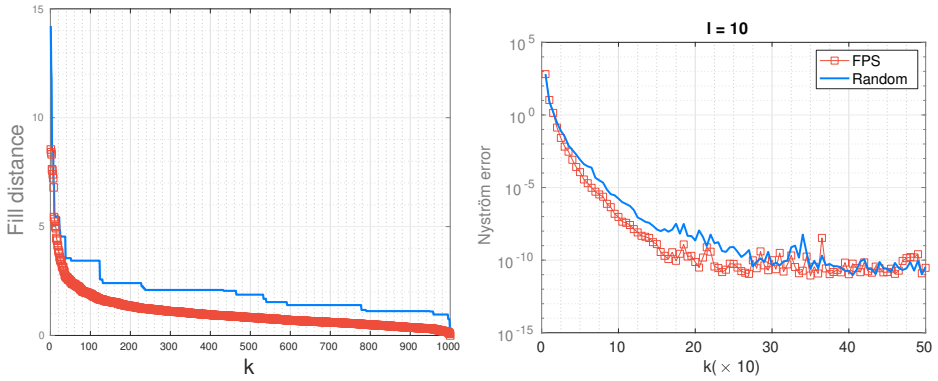

FIG. 4. *Comparison of fill distance and the Nyström approximation error for* 1000 *points uniformly sampled from a cube with edge length* 10*, when the Gaussian kernel function with length-scale* $l = 10$ *is used. FPS and random sampling are used to sample* k *points from* X *to form* X_k*. Nyström error is computed only for the ranks which are multiples of* 10.

the estimated rank is small. Meanwhile, the rank estimation algorithm discussed in section 3.3 also relies on FPS.

Remark 4.6. The error estimate is the upper bound in Theorem 4.5 does not involve the length-scale l explicitly. However, this error estimate can still help us understand how the length-scale in Gaussian kernels affects the Nyström approximation error when the same landmark points X_k are used. Assume h_{X_k} is the fill distance of X_k associated with the unit length-scale. When we change the length-scale to l, the kernel matrix associated with length-scale l can be regarded as a kernel matrix associated with the unit length-scale and the scaled data points $\tilde{\mathbf{x}} = \mathbf{x}/l$. This is because $\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\| = \|\frac{\mathbf{x}}{l} - \frac{\mathbf{y}}{l}\| = \frac{1}{l}\,\mathrm{dist}(\mathbf{x}, \mathbf{y})$. In this case, the fill distance on the rescaled data points becomes $\frac{h_{X_k}}{l}$. As a result, as l increases, the exponential factor in the estimate decays faster. This is consistent with the fact that the Gaussian kernel matrix $\mathbf{K}$ is numerically low rank when l is large.

4.4. FPS and screening effect. In this section, we delve deeper into the connection between FPS and the screening effect, providing further justification for the integration of FPS within the `AFN` framework. Empirical evidence in [24] supports FPS's superiority over alternative methods in numerous scenarios. The screening effect's effectiveness, closely tied to the uniformity of sampled points as measured by the ratio $\delta_X = \frac{q_X}{h_X}$, requires δ_X to be strictly bounded, both lower and upper, a condition met by FPS as proven in section 4.2. Further insights from section 3.2 in [40] and [39] indicate that FPS's selection of evenly distributed points enhances not only the sparsity in the Cholesky decomposition of the kernel matrix but also the precision matrix. Thus, FPS's application in sampling could increase the sparsity in $(\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$, improving the `AFN` preconditioner's efficiency and accuracy. We now demonstrate the screening effect (mentioned in section 3.1) numerically with an example in Figure 5 of when FPS is applied to select landmark points. Figure 5 shows histograms of the magnitude of the entries in three matrices $\mathbf{K}_{22} + \mu\mathbf{I}$, $\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12}$, and $(\mathbf{K}_{22} + \mu\mathbf{I} - \mathbf{K}_{12}^\top(\mathbf{K}_{11} + \mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$ for $l = 5$, with the matrices scaled so that their maximum entries are equal to one. The 1000 data points X are generated uniformly over a cube with edge length 10 and 100 landmark points X_{100} are selected

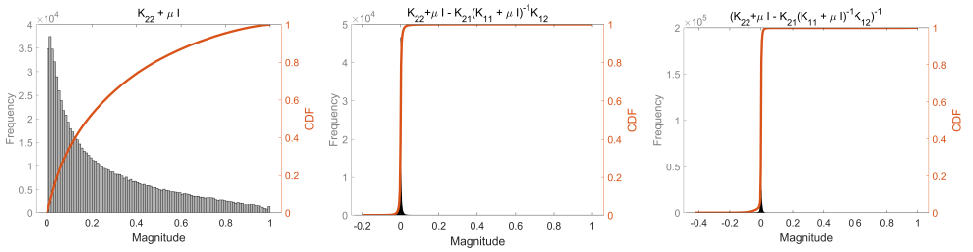

FIG. 5. *Histograms of the magnitude of the entries in* $\mathbf{K}_{22}+\mu\mathbf{I}$, $\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^{\top}(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12}$, *and* $(\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^{\top}(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$ *associated with a Gaussian kernel matrix defined using* 1000 *points sampled uniformly from a cube with edge length* 10, *regularization parameter* $\mu=0.0001$, *and length-scale* $l=5$. *The maximum entries in these three matrices are all scaled to* 1. $\mathbf{K}$ *has* 243 *eigenvalues greater than* $1.1\times\mu$.

Algorithm 4.1 Farthest point sampling (`FPS`).

1. **Input**: Dataset X of size n, number of samples k
2. **Output**: Landmark point set X_k of size k
3. Assign index to data points in X as $\mathbf{x}_1,\ldots,\mathbf{x}_n$
4. Calculate $\bar{\mathbf{x}}=\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$.
5. Set $l=\arg\min_{1\le j\le n}\operatorname{dist}(\mathbf{x}_j,\bar{\mathbf{x}})$
6. Initialize the set $X_k=\{\mathbf{x}_l\}$
7. Initialize the distance vector $\mathbf{d}$ of length n, setting all entries to inf
8. **for** $i=1$ to $k-1$ **do**
9. Update $\mathbf{d}$ with entries $\mathbf{d}(j)=\min\{\mathbf{d}(j),\operatorname{dist}(\mathbf{x}_j,\mathbf{x}_l)\}$
10. Choose $l=\arg\max_{1\le j\le n}\mathbf{d}(j)$
11. Add $\mathbf{x}_l$ to X_k
12. **end for**
13. **Return**: X_k

by FPS. The figure shows that $\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^{\top}(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12}$ and its inverse have many more entries with smaller magnitude than $\mathbf{K}_{22}+\mu\mathbf{I}$. This example further justifies that $(\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^{\top}(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$ has more "sparsity" than $\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^{\top}(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12}$, which supports the fact that FPS promotes sparsity in both kernel matrix and precision matrix and further supports the use of `FSAI` to $(\mathbf{K}_{22}+\mu\mathbf{I}-\mathbf{K}_{12}^{\top}(\mathbf{K}_{11}+\mu\mathbf{I})^{-1}\mathbf{K}_{12})^{-1}$.

4.5. Implementation of FPS. A straightforward implementation of FPS for selecting k samples from n points in $\mathbb{R}^d$ has a computational complexity of $O(dk^2n)$. However, this complexity can be reduced to $O(dkn)$ by maintaining a vector to track the distances between the unsampled points and the points already sampled. This optimization is elaborated in Algorithm 4.1.

5. Numerical experiments. The `AFN` preconditioner and the preconditioning strategy (Algorithm 3.5) are tested for the iterative solution of regularized kernel matrix systems (1.1) over a wide range of length-scale parameters l in the following two kernel functions:

- Gaussian kernel: $\mathcal{K}(\mathbf{x},\mathbf{y})=\exp\left(-\frac{1}{l^2}\|\mathbf{x}-\mathbf{y}\|_2^2\right)$.
- Matérn-3/2 kernel: $\mathcal{K}(\mathbf{x},\mathbf{y})=\left(1+\frac{\sqrt{3}}{l}||\mathbf{x}-\mathbf{y}||_2\right)\exp\left(-\frac{\sqrt{3}}{l}||\mathbf{x}-\mathbf{y}||_2\right)$.

We also benchmark the solution of these systems using unpreconditioned CG and preconditioned CG, with the FSAI preconditioner and the randomized Nyström (RAN) preconditioner [20] with randomly selected k landmark points.

RAN approximates the kernel matrix with a rank-k Nyström approximation based on randomly sampling the data points. Assuming the kth largest eigenvalue of $\mathbf{K}_{nys}$ is λ_k, the inverse of the RAN preconditioner takes the form [20] $(\lambda_k+\mu)\mathbf{U}(\mathbf{\Lambda}+\mu\mathbf{I})^{-1}\mathbf{U}^\top + (\mathbf{I}-\mathbf{U}\mathbf{U}^\top)$, where $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ is the eigendecomposition of $\mathbf{K}_{nys}$. In our experiments, we use 400 nearest neighbors as the sparsity pattern for FSAI, fix the Nyström rank to be 3000 for RAN, and use 100 nearest neighbors as the sparsity pattern for the FSAI used in AFN.

The stopping tolerance for the relative residual norm is set to be 10^{-4}. We randomly generated right-hand side vectors in equation (1.1) with entries from the uniform distribution $[-0.5, 0.5]$. For all tests we perform three runs and report the average results.

AFN, RAN, and FSAI have been implemented in C. The C implementation of the AFN preconditioner can be found in the AFN_Precond branch of the H2Pack GitHub website.[1] The test routines for AFN and RAN can be found from this webpage,[2] and the test routines for FSAI can be found from this webpage.[3] Experiments are run on a Ubuntu 20.04.4 LTS machine equipped with 755 GB of system memory and a 24-core 3.0 GHz Intel Xeon Gold 6248R CPU. We build our code with the GCC 9.4.0 compiler and take advantage of shared memory parallelism using OpenMP. We use the parallel BLAS and LAPACK implementation in the OpenBLAS library for basic matrix operations. H2Pack [9, 25] is used to provide linear complexity matrix-vector multiplications associated with large-scale $\mathbf{K}$ for 3D datasets with the relative error threshold 10^{-8}. We utilized a brute force parallel FPS algorithm on the global dataset. OpenMP was used to apply an $O(n)$ distance update in parallel at each step. The computational cost is tractable due to a maximum of 2000 distance updates required. The number of OpenMP threads is set to 24 in all the experiments.

5.1. Experiments with synthetic 3D datasets. The synthetic data consists of $n = 1.6 \times 10^5$ random points sampled uniformly from inside a 3D cube with edge length $\sqrt[3]{n}$. We first solve regularized linear systems associated with both Gaussian kernel and Matérn-3/2 kernel, with $\mu = 0.0001$.

The computational results are tabulated in Table 1, which shows the number of solver iterations required for convergence, the preconditioner setup (construction) time, and the time required for the iterative solve. Rank estimation Algorithm 3.4 is used to estimate the rank k for each kernel matrix, with the given length-scale information shown in the first row of Tables 1 and 3. For both kernels, we select nine *middle length-scales* to justify the robustness of AFN. We also include two extreme length-scales in the tables to show the effectiveness of the preconditioning strategy, using AFN summarized in Algorithm 3.5, across a wide range of l.

We first note that, for unpreconditioned CG, the iteration counts first increase and then decrease as the length-scale decreases for both kernel functions. This confirms the result seen earlier in Figure 1 that it is the linear systems associated with the *middle length-scales* that are most difficult to solve due to the unfavorable spectrum of these kernel matrices. We also observe that FSAI is very effective as a preconditioner for Gaussian kernel, with $l^2 = 0.1$ and Matérn-3/2 kernel, with $l = 1.0$. FSAI is effective

[1] https://github.com/scalable-matrix/H2Pack/.
[2] https://github.com/scalable-matrix/H2Pack/tree/AFN_precond/examples/AFN_precond.
[3] https://github.com/scalable-matrix/H2Pack/tree/AFN_precond/examples/SPDHSS-H2.

TABLE 1

Numerical results for the kernel matrices defined based on $n = 1.6 \times 10^5$ points sampled inside a 3D cube of edge length $\sqrt[3]{n}$. "–" indicates that a run failed to converge within 500 iterations. All experiments are run three times and reported as the average of three runs.

l^2	1000	65	60	55	50	45	40	35	30	25	0.1
k	565	9600	9600	9600	9600	12800	12800	12800	16000	19200	160000
Iteration Counts											
CG	44.00	-	-	-	-	-	-	-	-	-	1.00
AFN	3.00	35.00	37.00	38.00	40.00	42.00	46.00	50.00	57.00	62.00	1.00
RAN	3.00	72.67	101.33	140.67	199.33	284.33	409.33	-	-	-	-
FSAI	-	-	-	-	-	-	-	-	-	-	1.00
Setup Time (s)											
AFN	3.19	38.97	39.75	40.10	39.73	39.89	40.76	39.34	40.12	40.59	40.37
RAN	27.28	27.59	26.46	27.33	29.05	29.95	31.18	31.56	33.64	33.97	35.07
FSAI	10.00	9.91	10.02	10.16	9.72	9.87	10.14	9.71	10.01	9.84	13.22
Solve Time (s)											
CG	9.72	-	-	-	-	-	-	-	-	-	1.75
AFN	0.43	12.49	14.00	14.99	15.82	18.02	20.15	22.59	27.26	29.10	1.91
RAN	0.81	23.29	35.73	49.98	72.20	96.75	138.88	-	-	-	-
FSAI	-	-	-	-	-	-	-	-	-	-	1.27

(a) Gaussian kernel with a fixed $\mu = 0.0001$ and varying l.

$1/l$	1.0	0.065	0.060	0.055	0.050	0.045	0.040	0.035	0.030	0.025	0.001
k	160000	19200	16000	14080	12800	9600	9600	6400	6400	6400	178
Iteration Counts											
CG	293.67	-	-	-	-	-	-	-	-	-	292.67
AFN	3.00	6.00	6.00	6.00	7.00	7.00	7.00	7.00	7.00	6.00	9.00
RAN	-	454.00	404.33	355.67	308.33	263.00	220.67	181.00	142.00	108.33	4.00
FSAI	5.00	-	-	-	-	-	-	-	-	-	-
Setup Time (s)											
AFN	47.32	45.24	44.67	42.99	43.41	43.39	44.34	43.50	43.29	42.74	3.07
RAN	63.69	39.78	40.30	39.81	40.16	39.94	40.08	40.19	40.18	39.77	55.41
FSAI	13.98	10.31	10.18	10.19	10.29	10.26	10.30	10.28	10.02	9.84	13.80
Solve Time (s)											
CG	22.41	-	-	-	-	-	-	-	-	-	22.40
AFN	2.43	2.52	2.63	2.42	3.32	2.84	3.02	2.58	2.74	2.30	0.86
RAN	-	116.37	99.32	86.87	74.04	63.98	53.58	42.24	32.19	25.93	1.36
FSAI	3.71	-	-	-	-	-	-	-	-	-	-

(b) Matérn-3/2 kernel with a fixed $\mu = 0.0001$ and varying l.

if the inverse of the kernel matrix can be approximated by a sparse matrix, which is the situation for both length-scales. We observe the opposite effect for the RAN preconditioner, which is effective for large length-scales but poor for small length-scales. For middle length-scales, AFN substantially reduces the number of iterations compared to other methods. In particular, AFN yields almost a constant iteration number for Matérn-3/2 kernel. For Gaussian kernel with $l^2 = 1000$ and Matérn-3/2 kernel with $l = 1000$, choosing AFN as the Nyström preconditioner form with the estimated rank significantly reduces the setup time for AFN compared to RAN(3000) but still keeps roughly the same preconditioning effect.

In Table 2, we also compare the performance of AFN, RAN, and FSAI for solving (1.1) associated with the Matérn-3/2 kernel matrices with $l = 20$ and varying μ. It is easy to see that the performance of RAN and FSAI deteriorates as the regularization parameter μ decreases, while the iteration count of AFN remains almost a constant, which shows the improved robustness of AFN over RAN and FSAI with respect to μ.

TABLE 2

Numerical results for the Matérn-3/2 kernel matrices associated with $l = 20$ and varying μ and $n = 1.6 \times 10^5$ points sampled inside a 3D cube of edge length $\sqrt[3]{n}$. "–" indicates that a run failed to converge within 500 iterations. All experiments are run three times and reported as the average of three runs.

μ	1e−1	1e−2	1e−3	1e−4	1e−5	1e−6	1e−7	1e−8	1e−9	1e−10
Iteration Counts										
CG	-	-	-	-	-	-	-	-	-	-
AFN	15.00	12.00	6.00	7.00	7.00	7.00	7.00	7.00	7.00	7.00
RAN	10.33	29.00	93.33	311.33	-	-	-	-	-	–
FSAI	164.00	370.33	-	-	-	-	-	-	-	-
Setup Time (s)										
AFN	43.74	43.50	42.74	44.59	43.63	43.24	44.31	44.30	43.11	43.71
RAN	40.25	39.71	39.14	40.86	39.92	40.13	40.40	40.34	39.80	40.35
FSAI	10.33	10.46	10.56	10.39	10.53	10.40	10.53	10.59	10.76	10.48
Solve Time (s)										
CG	-	-	-	-	-	-	-	-	-	-
AFN	5.30	4.95	2.61	2.78	3.02	2.90	2.89	2.84	2.88	3.09
RAN	3.29	8.53	25.03	76.33	-	-	-	-	-	-
FSAI	21.43	46.44	-	-	-	-	-	-	-	-

5.2. Experiments with machine learning datasets. In this section we test the performance of AFN on two high-dimensional datasets, namely IJCNN1 from LIBSVM [11] and Elevators from UCI [27]. The training set of IJCNN1 consists of $n = 49990$ data points, with 22 features and 2 classes, while Elevators contains $n = 16599$ data points, with 18 features and 1 target.

Here, we perform experiments with the Gaussian kernel for IJCNN1 and Matérn-3/2 kernel for Elevators. After conducting grid searches, we select the regularization parameter to be $\mu = n \times 10^{-6}$ for both datasets so that the test error of KRR is small for the optimal length-scale l in our searches. We select 12 length-scales in two separate intervals, which include the optimal length-scales for both datasets. The grid search method was used to determine the optimal length-scale for IJCNN1, resulting in a value of $l = 1$ which is consistent with the findings in [20]. In contrast, for Elevators, the optimal length-scale was determined using GPyTorch [50] and found to be $l = 14$. Most of the length-scales within each interval correspond to middle length-scales. Two extreme length-scales are also considered here to show the effectiveness of AFN across a wide range of l. Since FSAI is less robust than RAN, we only compare AFN with RAN in this section. As these are moderate-dimensional datasets (22 and 18 dimensions, as mentioned) and we do not have a fast kernel matrix-vector multiplication code for these datasets, the kernel matrix-vector multiplications were performed explicitly. Due to the high computational cost of FPS in high dimensions, we simply use uniform sampling to select the landmark points for AFN when the estimated rank is greater than 2000 in these experiments.

We report the computational results in Table 3. The patterns of the change of iteration counts, setup time, and solution time with respect to the length-scales on both datasets are similar to those observed in the 3D experiments. First, the iteration counts of unpreconditioned CG first increase and then decrease as l decreases in both datasets. This indicates that the spectrum of the kernel matrices associated with high-dimensional datasets could be related to those associated with low-dimensional data. AFN is again able to significantly reduce the iteration counts compared to unpreconditioned CG in all tests. We note that the iteration count of the RAN preconditioned

TABLE 3

Numerical results for the IJCNN1 and Elevator datasets with Gaussian kernel and Matérn-3/2 kernel, respectively. "–" indicates that a run failed to converge within 500 iterations. All experiments are run three times and reported as the average of three runs. In both tests we set $\mu = n \times 10^{-6}$.

l^2	10.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.01
k	1278	8798	10397	11197	13197	14996	17396	20395	24394	29394	37192	48190
Iteration Counts												
CG	218.00	-	-	-	-	-	-	-	-	481.00	418.00	239.00
AFN	3.00	44.00	43.33	42.00	41.00	39.00	36.67	33.00	29.33	25.33	19.67	9.00
RAN	2.00	12.67	13.67	15.67	18.67	21.67	26.00	32.00	40.00	51.00	66.67	73.33
Setup Time (s)												
AFN	4.18	15.69	15.66	15.30	15.53	15.29	15.30	15.68	16.34	15.51	15.19	15.15
RAN	52.44	40.81	41.68	41.20	41.73	41.40	41.09	41.59	41.08	40.90	43.58	48.16
Solve Time (s)												
CG	30.63	-	-	-	-	-	-	-	-	55.23	46.73	34.73
AFN	0.97	8.07	8.99	8.24	7.47	7.55	6.88	6.50	5.94	5.05	5.01	2.44
RAN	0.70	2.93	3.04	3.01	4.03	4.90	4.87	6.13	8.11	9.40	11.89	12.83

(a) IJCNN1 with Gaussian kernel.

$1/l$	1.0	0.1	0.09	0.08	0.07	0.06	0.05	0.04	0.03	0.02	0.01	0.0005
k	16599	12083	11685	11419	11087	10822	10224	9427	8166	6838	5576	983
Iteration Counts												
CG	29.00	324.00	325.00	331.00	339.00	347.00	355.00	358.00	349.00	331.00	303.00	124.00
AFN	3.00	9.33	9.67	9.67	10.00	10.00	10.00	10.00	10.00	49.00	60.00	5.00
RAN	20.67	71.67	71.00	69.33	67.00	65.00	61.00	57.33	59.67	69.67	75.33	7.33
Setup Time (s)												
AFN	9.58	5.34	5.45	5.79	5.60	5.48	5.42	5.47	5.36	5.76	6.06	1.94
RAN	38.78	28.64	44.28	42.45	30.86	32.53	44.61	36.91	39.38	38.32	35.72	34.90
Solve Time (s)												
CG	0.54	3.65	3.73	3.71	3.79	3.92	4.01	4.06	3.93	3.75	3.48	1.39
AFN	0.21	0.38	0.40	0.43	0.40	0.40	0.49	0.39	0.38	1.83	2.22	0.11
RAN	0.68	2.04	1.84	2.08	1.82	1.76	1.67	1.49	1.76	1.88	2.00	0.28

(b) Elevators with Matérn-3/2 kernel.

CG increases as the estimated rank increases on the IJCNN1 dataset. This implies that in order to converge in the same number of iterations as l becomes smaller, RAN-type preconditioners need to keep increasing the Nyström approximation rank k and thus require a longer setup time and more storage. AFN requires a shorter setup time in all of the experiments and leads to smaller iteration counts when $l^2 < 0.4$ on the IJCNN1 dataset and all length-scales on the Elevators dataset. In addition, we can also observe that AFN yields the smallest total time in all of the experiments on both datasets compared with RAN.

6. Conclusion. In this paper, we introduced an approximate block factorization of $\mathbf{K} + \mu\mathbf{I}$ that is inspired by the existence of a Nyström approximation $\mathbf{K} \approx \mathbf{K}_{X,X_k}\mathbf{K}_{X_k,X_k}^{-1}\mathbf{K}_{X_k,X}$. The approximation is designed to efficiently handle the case where k is large by using sparse approximate inverses.

We further introduced a preconditioning strategy that is robust for a wide range of length-scales. When the length-scale is large, existing Nyström preconditioners work well. For the challenging length-scales, the AFN preconditioner proposed in this paper is the most effective. We justify the use of FPS to select landmark points in order to

construct an accurate and stable AFN preconditioner and propose a rank estimation algorithm using a subsampling of the entire dataset.

It is important to note that in high-dimensional settings, the effectiveness of screening effects diminishes, as indicated by [39, 40]. This is attributed to the reduced representational capacity of Euclidean distance for spatial similarity in high-dimensional spaces, a concept further explored by [15]. Consequently, the FSAI approach for approximating the inverse of the Schur complement can be less effective for high-dimensional datasets, such as those commonly found in machine learning, as it is for lower-dimensional ones, such as those in spatial statistics. Nevertheless, in the realm of machine learning, kernel methods—including the kernel trick in support vector, machines (SVMs), kernel ridge regression (KRR), and Gaussian process regression (GPR)—fundamentally rely on the premise that spatial similarity correlates with data similarity, and the proposed AFN method retains its relevance as long as this assumption is valid. For datasets with high dimensionality, we plan to first apply a transformation to map the data points to lower-dimensional manifolds. This transformation, as discussed in the survey [6], ensures that Euclidean distance continues to effectively represent similarity in these reduced-dimensional spaces. In future work, we will also study whether the dependence on ambient dimension in Theorem 4.3 can be reduced to the intrinsic dimension of the data manifold and apply AFN to accelerate the convergence of stochastic trace estimation and gradient-based optimization algorithms.

Appendix A. Proof of Theorem 4.5. The proof of Theorem 4.5 relies on Theorem A from [5]. Our Theorem A.1 states that any bounded map $\mathcal{T}$ from a Hilbert space to an RKHS $\mathcal{H}$ corresponding to certain smooth radial kernels, such as the Gaussian kernel defined in (1.2) and the inverse multiquadrics kernel defined in (4.9), always admits a low-rank approximation in $L^2_\mu := \{f(x) | \int |f(x)|^2 d\mu < \infty\}$. Furthermore, the approximation error bound can be quantified by fill distance. Before we proceed to Theorem A.1, we first introduce some notation that will be used in the statement of Theorem A.1. On a domain Ω, the integral operator $\mathcal{K}_\mu : L^2_\mu \to \mathcal{H}$ is defined as

$$\mathcal{K}_\mu(f)(\cdot) = \int \mathcal{K}(\cdot, \mathbf{x}) f(\cdot) d\mu.$$

The restriction operator $\mathcal{R}_\mu : \mathcal{H} \to L^2_\mu$ is defined as the restriction of $f \in \mathcal{H}$ to the support of μ, and interpolation operator $\mathcal{S}_{X_k} : \mathcal{H} \to \mathcal{H}$ is defined by interpolating the values of f on a subset $X_k \subset \Omega$ as

$$\mathcal{S}_{X_k}(f)(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}),$$

with $(\alpha_1, \ldots, \alpha_k)^\top = \mathbf{K}^{-1}_{X_k, X_k} (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k))^\top$. Since the ranges of $\mathcal{R}_\mu$ and $\mathcal{S}_{X_k}$ are different, the following norm is used to measure their difference:

$$\|\mathcal{R}_\mu - \mathcal{S}_{X_k}\|_{\mathcal{H} \to L^2_\mu} := \max_{f \in \mathcal{H}, f \neq 0} \frac{\|(\mathcal{R}_\mu - \mathcal{S}_{X_k})(f)\|_{L^2_\mu}}{\|f\|_{\mathcal{H}}}.$$

THEOREM A.1 (see [5, Theorem A]). *Let $\mathcal{H}$ denote the RKHS corresponding to the kernel $\mathcal{K}$. Given a probability measure μ on Ω and a set $X_k \subset \Omega$, there exist constants C', $C'' > 0$ such that*

$$\|\mathcal{R}_\mu - \mathcal{S}_{X_k}\|_{\mathcal{H}\to L^2_\mu} < C' \exp(-C''/h_{X_k}). \tag{A.1}$$

When $\Omega = X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and the uniform discrete measure $\mu_X = \frac{1}{n}\sum_{i=1}^n \delta_{\mathbf{x}_i}$ is used with $\delta_{\mathbf{x}_i}$ being the Dirac measure at point $\mathbf{x}_i$, we have

$$\mathcal{K}_{\mu_X}(f)(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^n \mathcal{K}(\mathbf{x}_i, \mathbf{x}) f(\mathbf{x}_i)$$

and $\mathcal{H}_X = \operatorname{span}\{K(\mathbf{x}_1, \cdot), \ldots, K(\mathbf{x}_n, \cdot)\}$. The integral operator, interpolation operator, and restriction operator can then be written in the matrix form as $\mathcal{K}_{\mu_X}(f)(X) = \frac{1}{n}\mathbf{K} f(X)$, $\mathcal{S}_{X_k}(f)(X) = \mathbf{K}_{X,X_k}\mathbf{K}^{-1}_{X_k,X_k} f(X_k)$, and $\mathcal{R}_{\mu_X} = \mathbf{I} \in \mathbb{R}^{n\times n}$, respectively. Since $\mathcal{R}_{\mu_X} \circ \mathcal{K}_{\mu_X} = \mathcal{K}_{\mu_X}$, we have

$$\mathcal{K}_{\mu_X} - \mathcal{S}_{X_k} \circ \mathcal{K}_{\mu_X} = (\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}.$$

Thus, we can get the following inequality:

$$\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to L^2_{\mu_X}} \le \|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k})\|_{\mathcal{H}_X \to L^2_{\mu_X}} \|\mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to \mathcal{H}_X}.$$

Based on Theorem A.1, we know that

$$\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to L^2_{\mu_X}} \le C' \exp(-C''/h_{X_k}) \|\mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to \mathcal{H}_X}.$$

In the next theorem, we will derive an error estimate for the Nyström approximation error by further proving

$$\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to L^2_{\mu_X}} = \frac{1}{n}\|\mathbf{K} - \mathbf{K}_{nys}\|$$

and $\|\mathcal{K}_{\mu_X}\|^2_{L^2_{\mu_X}\to \mathcal{H}_X} = \sqrt{\|\mathbf{K}\|/n}$.

THEOREM 4.5. *The Nyström approximation* $\mathbf{K}_{nys} = \mathbf{K}_{X,X_k}\mathbf{K}^{-1}_{X_k,X_k}\mathbf{K}_{X_k,X}$ *to* $\mathbf{K}$ *using the landmark points* $X_k = \{\mathbf{x}_{k_i}\}_{i=1}^k$ *has the error estimate*

$$\|\mathbf{K} - \mathbf{K}_{nys}\| < \sqrt{n\|\mathbf{K}\|} C' \exp(-C''/h_{X_k}), \tag{A.2}$$

where C' *and* C'' *are constants independent of* X_k.

Proof. Since $\mathcal{R}_{\mu_X} \circ \mathcal{K}_{\mu_X} = \mathcal{K}_{\mu_X}$, we have

$$\mathcal{K}_{\mu_X} - \mathcal{S}_{X_k} \circ \mathcal{K}_{\mu_X} = (\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}.$$

Notice that $\mathcal{K}_{\mu_X}$ is a map from $L^2_{\mu_X}$ to $\mathcal{H}_X$, and from the definition of the norm, we get the following inequality:

$$\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to L^2_{\mu_X}} \le \|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k})\|_{\mathcal{H}_X \to L^2_{\mu_X}} \|\mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to \mathcal{H}_X}. \tag{A.3}$$

Based on Theorem A.1, we obtain

$$\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k})\|_{\mathcal{H}_X \to L^2_{\mu_X}} < C' \exp(-C''/h_{X_k}). \tag{A.4}$$

First, recall that

$$\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to L^2_{\mu_X}} = \max_{f\in L^2_{\mu_X}, f\neq 0} \frac{\|(\mathcal{R}_{\mu_X} - \mathcal{S}_{X_k}) \circ \mathcal{K}_{\mu_X}(f)\|_{L^2_{\mu_X}}}{\|f\|_{L^2_{\mu_X}}}$$

and

$$\begin{aligned}\|(\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}(f)\|_{L^2_{\mu_X}} &= \sqrt{\int_X \left((\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}(f)\right)^2 d\mu_X} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^n \left((\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}(f)(\mathbf{x}_i)\right)^2} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^n \left((\mathcal{R}_{\mu_X}\circ\mathcal{K}_{\mu_X}(f)(\mathbf{x}_i)-\mathcal{S}_{X_k}\circ\mathcal{K}_{\mu_X}(f)(\mathbf{x}_i))\right)^2}.\end{aligned}$$

Define two vectors based on the two function evaluations at X:

$$\mathbf{F}_1=(\mathcal{R}_{\mu_X}\circ\mathcal{K}_{\mu_X}(f))(X), \quad \text{and} \quad \mathbf{F}_2=(\mathcal{S}_{X_k}\circ\mathcal{K}_{\mu_X}(f))(X).$$

Then we obtain

$$\|(\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}(f)\|_{L^2_{\mu_X}}=\frac{1}{\sqrt{n}}\|\mathbf{F}_1-\mathbf{F}_2\|.$$

Notice that $\mathbf{F}_1$ and $\mathbf{F}_2$ can also be written as

$$\mathbf{F}_1=\frac{1}{n}\mathbf{K}f(X) \quad \text{and} \quad \mathbf{F}_2=\frac{1}{n}\mathbf{K}_{X,X_k}\mathbf{K}^{-1}_{X_k,X_k}\mathbf{K}_{X_k,X}f(X).$$

Thus,

$$\begin{aligned}\|(\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}(f)\|_{L^2_{\mu_X}} &= \frac{1}{\sqrt{n}}\|\frac{1}{n}\mathbf{K}f(X)-\frac{1}{n}\mathbf{K}_{X,X_k}\mathbf{K}^{-1}_{X_k,X_k}\mathbf{K}_{X_k,X}f(X)\| \\ &= \frac{1}{n^{3/2}}\|(\mathbf{K}-\mathbf{K}_{nys})f(X)\|.\end{aligned}$$

On the other hand,

$$\|f\|_{L^2_{\mu_X}}=\sqrt{\int_X f^2 d\mu_X}=\sqrt{\frac{1}{n}\sum_{i=1}^n f(\mathbf{x}_i)^2}=\frac{1}{\sqrt{n}}\|f(X)\|.$$

As a result, we get

$$\begin{aligned}\|(\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}\|_{L^2_{\mu_X}\to L^2_{\mu_X}} &= \max_{f\in L^2_{\mu_X},f\neq 0}\frac{\|(\mathcal{R}_{\mu_X}-\mathcal{S}_{X_k})\circ\mathcal{K}_{\mu_X}(f)\|_{L^2_{\mu_X}}}{\|f\|_{L^2_{\mu_X}}} \\ &= \max_{f\in L^2_{\mu_X},f\neq 0}\frac{\|(\mathbf{K}-\mathbf{K}_{nys})f(X)\|}{n\|f(X)\|} \\ &= \max_{\mathbf{f}\in\mathbb{R}^n,\mathbf{f}\neq 0}\frac{\|(\mathbf{K}-\mathbf{K}_{nys})\mathbf{f}\|}{n\|\mathbf{f}\|}=\frac{1}{n}\|\mathbf{K}-\mathbf{K}_{nys}\|.\end{aligned}$$

Since there exists an orthogonal basis $\{f_i\}_{i=1}^n$ of eigenfunctions of $\mathcal{K}_{\mu_X}$ in $L^2_{\mu_X}$ with the eigenvalues λ_i, we can express any $f\in L_{\mu_X^2}$ as $f=\sum_{i=1}^n\alpha^{(i)}f_i$. As a result, we have

$$\begin{aligned}\|\mathcal{K}_{\mu_X}\|^2_{L^2_{\mu_X}\to\mathcal{H}_X} &= \max_{f\in L^2_{\mu_X},f\neq 0}\frac{\|\mathcal{K}_{\mu_X}(f)\|^2_{\mathcal{H}_X}}{\|f\|^2_{L^2_{\mu_X}}} \\ &= \max_{f\in L^2_{\mu_X},f\neq 0}\frac{\langle\sum_{i=1}^n\alpha^{(i)}\mathcal{K}_{\mu_X}(f_i),\sum_{i=1}^n\alpha^{(i)}\mathcal{K}_{\mu_X}(f_i)\rangle_{\mathcal{H}_X}}{\|\sum_{i=1}^n\alpha^{(i)}f_i\|^2_{L^2_{\mu_X}}}.\end{aligned}$$

Proposition 10.28 in [48] shows that $\{\mathcal{K}_{\mu_X}(f_i)\}$ is orthogonal in $\mathcal{H}_X$:

$$\langle \mathcal{K}_{\mu_X}(f_i), \mathcal{K}_{\mu_X}(f_j)\rangle_{\mathcal{H}_X} = \langle \mathcal{R}_{\mu_X}\mathcal{K}_{\mu_X}(f_i), f_j\rangle_{L^2_{\mu_X}} = \lambda_i \langle f_i, f_j\rangle_{L^2_{\mu_X}}. \tag{A.5}$$

Thus we obtain

$$\begin{aligned}
\|\mathcal{K}_{\mu_X}\|^2_{L^2_{\mu_X}\to\mathcal{H}_X} &= \max_{f\in L^2_{\mu_X}, f\neq 0} \frac{\sum_{i=1}^n \lambda_i |\alpha^{(i)}|^2 \|f_i\|^2_{L^2_{\mu_X}}}{\sum_{i=1}^n |\alpha^{(i)}|^2 \|f_i\|^2_{L^2_{\mu_X}}} \\
&= \max_{f\in L^2_{\mu_X}, f\neq 0} \sum_{i=1}^n \frac{|\alpha^{(i)}|^2 \|f_i\|^2_{L^2_{\mu_X}}}{\sum_{i=1}^n |\alpha^{(i)}|^2 \|f_i\|^2_{L^2_{\mu_X}}} \lambda_i \\
&= \lambda_1.
\end{aligned}$$

Since

$$\mathcal{K}_{\mu_X}(f_i)(X) = \lambda_i f_i(X) \quad \text{and} \quad \mathcal{K}_{\mu_X}(f_i)(X) = \frac{1}{n}\mathbf{K} f_i(X),$$

we get

$$\mathbf{K} f_i(X) = n\lambda_i f_i(X),$$

which implies that $n\lambda_i$ are the eigenvalues of the kernel matrix $\mathbf{K}$, and in particular,

$$n\lambda_1 = \|\mathbf{K}\|. \tag{A.6}$$

Finally, we have

$$\frac{1}{n}\|\mathbf{K} - \mathbf{K}_{nys}\| < \sqrt{\lambda_1} C' \exp(-C''/h_{X_k}) = \frac{1}{\sqrt{n}} \sqrt{\|\mathbf{K}\|} C' \exp(-C''/h_{X_k}). \qquad \square$$

Reproducibility of computational results. This paper has been awarded the "SIAM Reproducibility Badge: Code and data available" as a recognition that the authors have followed reproducibility principles valued by SISC and the scientific computing community. Code and data that allow readers to reproduce the results in this paper are available at https://github.com/scalable-matrix/H2Pack/tree/AFN_precond and in the supplementary materials (H2Pack.zip [local/web 3.99MB]).

Acknowledgments. The authors thank Zachary Frangella for sharing his MATLAB implementation of the randomized Nyström preconditioner [20]. The authors also thank two anonymous referees for their valuable suggestions, which greatly improved the presentation of the paper.

REFERENCES

[1] A. Alaoui and M. W. Mahoney, *Fast randomized kernel ridge regression with statistical guarantees*, in Advances in Neural Information Processing Systems 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds., Curran Associates, 2015.

[2] S. Ambikasaran and E. Darve, *An $\mathcal{O}(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices: With application to radial basis function interpolation*, J. Sci. Comput., 57 (2013), pp. 477–501.

[3] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil, *Fast direct methods for Gaussian processes*, IEEE Trans. Pattern Anal. Mach. Intell., 38 (2016), pp. 252–265, https://doi.org/10.1109/TPAMI.2015.2448083.

[4] M.-A. Belabbas and P. J. Wolfe, *Spectral methods in machine learning and new strategies for very large datasets*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 369–374.

[5] M. Belkin, *Approximation beats concentration? An approximation view on inference with smooth radial kernels*, in Proceedings of the 31st Conference On Learning Theory, Proc. Mach. Learn. Res. 75, PMLR, 2018, pp. 1348–1361.
[6] M. Binois and N. Wycoff, *A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization*, ACM Trans. Evol. Learn. Optim., 2 (2022), pp. 1–26.
[7] D. Cai, E. Chow, L. Erlandson, Y. Saad, and Y. Xi, *SMASH: Structured matrix approximation by separation and hierarchy*, Numer. Linear Algebra Appl., 25 (2018), e2204.
[8] D. Cai, E. Chow, and Y. Xi, *Data-driven linear complexity low-rank approximation of general kernel matrices: A geometric approach*, Numer. Linear Algebra Appl., 30 (2023), e2519.
[9] D. Cai, H. Huang, E. Chow, and Y. Xi, *Data-driven construction of hierarchical matrices with nested bases*, SIAM J. Sci. Comput., 46 (2024), pp. S24–S50.
[10] D. Cai, J. Nagy, and Y. Xi, *Fast deterministic approximation of symmetric indefinite kernel matrices with high dimensional datasets*, SIAM J. Matrix Anal. Appl., 43 (2022), pp. 1003–1028, https://doi.org/10.1137/21M1424627.
[11] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intell. Syst. Technol. (TIST), 2 (2011), pp. 1–27.
[12] Y. Chen, E. N. Epperly, J. A. Tropp, and R. J. Webber, *Randomly Pivoted Cholesky: Practical Approximation of a Kernel Matrix with Few Entry Evaluations*, preprint, arXiv:2207.06503, 2022.
[13] D. Y. Chenhan, S. Reiz, and G. Biros, *Distributed O(n) linear solver for dense symmetric hierarchical semi-separable matrices*, in Proceedings of the 2019 IEEE 13th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip (MCSoC), IEEE, 2019, pp. 1–8.
[14] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, *Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets*, J. Amer. Statist. Assoc., 111 (2016), pp. 800–812.
[15] P. Domingos, *A few useful things to know about machine learning*, Commun. ACM, 55 (2012), pp. 78–87.
[16] P. Drineas and M. W. Mahoney, *On the Nyström method for approximating a Gram matrix for improved kernel-based learning*, J. Mach. Learn. Res., 6 (2005), pp. 2153–2175.
[17] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, *The farthest point strategy for progressive image sampling*, IEEE Trans. Image Process., 6 (1997), pp. 1305–1315.
[18] L. Erlandson, D. Cai, Y. Xi, and E. Chow, *Accelerating parallel hierarchical matrix-vector products via data-driven sampling*, in Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, 2020, pp. 749–758, https://doi.org/10.1109/IPDPS47924.2020.00082.
[19] G. E. Fasshauer, *Meshfree Approximation Methods with Matlab*, Interdiscip. Math. Sci. 6, World Scientific, 2007, https://doi.org/10.1142/6437.
[20] Z. Frangella, J. A. Tropp, and M. Udell, *Randomized Nyström preconditioning*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 718–752, https://doi.org/10.1137/21M1466244.
[21] A. Gittens and M. W. Mahoney, *Revisiting the Nyström method for improved large-scale machine learning*, J. Mach. Learn. Res., 17 (2016), pp. 3977–4041.
[22] T. F. Gonzalez, *Clustering to minimize the maximum intercluster distance*, Theoret. Comput. Sci., 38 (1985), pp. 293–306.
[23] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 79–94, https://doi.org/10.1137/0912004.
[24] J. Guinness, *Permutation and grouping methods for sharpening Gaussian process approximations*, Technometrics, 60 (2018), pp. 415–429.
[25] H. Huang, X. Xing, and E. Chow, *H2Pack: High-performance H^2 matrix package for kernel matrices using the proxy point method*, ACM Trans. Math. Softw., 47 (2020), pp. 1–29, https://doi.org/10.1145/3412850.
[26] M. Katzfuss and J. Guinness, *A General Framework for Vecchia Approximations of Gaussian Processes*, preprint, https://arxiv.org/abs/1708.06302, 2019.
[27] M. Kelly, R. Longjohn, and K. Nottingham, *The UCI Machine Learning Repository*, https://archive.ics.uci.edu.
[28] L. Y. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58, https://doi.org/10.1137/0614004.
[29] D. Lazzaro and L. B. Montefusco, *Radial basis functions for the multivariate interpolation of large scattered data sets*, J. Comput. Appl. Math., 140 (2002), pp. 521–536.

[30] C. Li, S. Jegelka, and S. Sra, *Fast DPP sampling for Nyström with application to kernel methods*, in Proceedings of the 33rd International Conference on Machine Learning, New York, NY, JMLR: W&CP 48, PMLR, 2016, pp. 2061–2070.
[31] W. B. March, B. Xiao, S. Tharakan, C. D. Yu, and G. Biros, *Robust treecode approximation for kernel machines*, in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 775–784.
[32] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), pp. 403–572, https://doi.org/10.1017/S0962492920000021.
[33] S. Müller, *Komplexität und Stabilität von kernbasierten Rekonstruktionsmethoden*, Ph.D. thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, 2009.
[34] C. Musco and C. Musco, *Recursive sampling for the Nyström method*, in Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, Adv. Neural Inf. Process. Syst. 30, Curran Associates, 2017 pp. 3836–3848.
[35] G. Peyré and L. D. Cohen, *Geodesic remeshing using front propagation*, Int. J. Comput. Vision, 69 (2006), pp. 145–156.
[36] M. Pourahmadi, *Joint mean-covariance models with applications to longitudinal data: Unconstrained parameterisation*, Biometrika, 86 (1999), pp. 677–690.
[37] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning,* MIT Press, 2005.
[38] E. Rebrova, G. Chávez, Y. Liu, P. Ghysels, and X. S. Li, *A study of clustering techniques and hierarchical matrix formats for kernel ridge regression*, in Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2018, pp. 883–892.
[39] F. Schäfer, M. Katzfuss, and H. Owhadi, *Sparse Cholesky factorization by Kullback–Leibler minimization*, SIAM J. Sci. Comput., 43 (2021), pp. A2019–A2046, https://doi.org/10.1137/20M1336254.
[40] F. Schäfer, T. J. Sullivan, and H. Owhadi, *Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity*, Multiscale Model. Simul., 19 (2021), pp. 688–730, https://doi.org/10.1137/19M129526X.
[41] T. Schlömer, D. Heck, and O. Deussen, *Farthest-point optimized point sets with maximized minimum distance*, in Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, Amer. Math. Soc., 2011, pp. 135–142.
[42] G. Shabat, E. Choshen, D. B. Or, and N. Carmel, *Fast and accurate Gaussian kernel ridge regression using matrix decompositions for preconditioning*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 1073–1095, https://doi.org/10.1137/20M1343993.
[43] S. Si, C.-J. Hsieh, and I. Dhillon, *Memory efficient kernel approximation*, J. Mach. Learn. Res., 18 (2017), pp. 1–32.
[44] M. L. Stein, *The screening effect in kriging*, Ann. Statist., 30 (2002), pp. 298–323.
[45] M. L. Stein, 2010 *Rietz lecture: When does the screening effect hold?*, Ann. Statist., 39 (2011), pp. 2795–2819.
[46] M. L. Stein, *When does the screening effect not hold?*, Spat. Stat., 11 (2015), pp. 65–80.
[47] A. V. Vecchia, *Estimation and model identification for continuous spatial processes*, J. R. Stat. Soc. Ser. B Methodol., 50 (1988), pp. 297–312.
[48] H. Wendland, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math. 17, Cambridge University Press, 2004.
[49] H. Wendland, *Computational aspects of radial basis function approximation*, Stud. Comput. Math., 12 (2006), pp. 231–256.
[50] J. Wenger, G. Pleiss, P. Hennig, J. Cunningham, and J. Gardner, *Preconditioning for scalable Gaussian process hyperparameter optimization*, in Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, PMLR 162, 2022, pp. 23751–23780.
[51] D. J. White, *The maximal-dispersion problem*, IMA J. Manag. Math., 3 (1991), pp. 131–140.
[52] C. K. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, in Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 682–688.
[53] Y. Wu, *Packing, Covering, and Consequences on Minimax Risk*, Course Lecture Notes for ECE598: Information-Theoretic Methods for High-Dimensional Statistics, Yale University, 2016.

[54] C. Yang, R. Duraiswami, and L. S. Davis, *Efficient kernel machines using the improved fast Gauss transform*, in Advances in Neural Information Processing Systems 17, L. Saul, Y. Weiss, and L. Bottou, eds., MIT Press, 2004, https://proceedings.neurips.cc/paper/2004/file/85353d3b2f39b9c9b5ee3576578c04b7-Paper.pdf.

[55] K. Zhang and J. T. Kwok, *Clustered Nyström method for large scale manifold learning and dimension reduction*, IEEE Trans. Neural Netw., 21 (2010), pp. 1576–1587.